Automata, Complexity, and Optimality Theory

Gerhard Jäger & Stephan Kepser

Abstract

Bob Frank, Giorgio Satta, and Lauri Karttunen stunned the computational OT community some years ago when they pointed out that unidirectional optimization is essentially a finite state technique. If all components of an OT system can be modeled by a finite state machine, then the complexity of the entire system does not exceed the complexity of its components.

There has been some debate in past years whether simple unidirectional optimization should be complemented by some component of "recoverability", which led to various notions of bidirectional optimization. The most radical of these proposals, Blutner's "Weak Bidirectionality", is in fact computationally more complex than all other proposed evaluation methods because it introduces an element of recursion.

Recent work by Christian Wartena, Stephan Kepser and Uwe Mönnich shows that the complexity landscape for finite state OT is almost identicallly replicated at the level of tree automata. This class of automata is considerably more expressive than finite state string automata. Therefore the new results promise to be relevant for the working OT syntactician/semanticist.

After a recapitulation of previous work on the automata theoretic complexity of unidirectional and bidirectional OT, the paper explores the implementability of bidirectional OT with finite state tree automata.

1 Introduction

Most versions of generative grammar are localist theories. They assume that that grammatical status of a given linguistic item is exclusively determined by its internal structure. In contrast to this, Optimality Theory (OT henceforth) is a holistic theory. Whether a certain structure is grammatical depends on its internal structure (whether it is licensed by the generator), but it also depends on its relation to other linguistics items, the competing candidates. There are good reasons to believe that natural languages are indeed holistic systems in this sense, and that OT thus formalizes an important facette of the truth.

While the non-localist nature of OT is certainly fascinating and insightful, it seems to make OT grammars computationally intractable. If a potentially infinite set of competing candidates has to be evaluated just to parse or to generate a single word or sentence, something seems to be wrong. This, however, is only an argument against the most naive version of the OT evaluation algorithm, namely brute force *generate and test*. Under

the connectionist interpretation of OT, sub-optimal candidates are never generated, and the naive evaluation procedure is just a coarse symbolic description of massively parallel harmony maximization on the sub-symbolic level.

Alternatively, OT evaluation can be made tractable if the evaluator operates on compact *descriptions* of candidate sets, rather than on the candidates themselves. A few years ago, Frank and Satta (1998) and Karttunen (1998) stunned the computational OT community with constructions that proved that OT evaluation is not just tractable, it is in fact implementable with finite state automata. These are the most efficient automata that are in use in computational linguistics. Their crucial insight was that the generator of an OT system can be modeled as a rational relation, which can in turn be implemented as a finite state transducer. Likewise, constraints can be identified with the set of candidates that obey them. If each constraint, viewed as a set, turns out to be a regular language, there is an constructive proof of the existence of a finite state transducer that implements the corresponding OT evaluator.

Frank and Satta's construction implements unidirectional optimization of regular languages and relations. It has inspired a series of generalizations and modifications during the past few years. We can distinguish two lines of research here:

- Regular languages and relations are by and large appropriate for phonology and morphology, but they are too restricted for syntax and semantics. Various results by Wartena (2000), Kepser and Mönnich (2003) show though that very similar constructions can be applied to various classes of tree languages and relations.

- Various people, notably Blutner (2000), have argued that for semantics and pragmatics, bidirectional optimization is more appropriate than unidirectional optimization. Jäger (2002) generalizes Frank and Satta's result to Blutner's notion of (weak) bidirectional OT.

In this paper, we attempt to combine these two strands of research. After briefly reviewing the mentioned results, we will suggest two constructions of bidirectional optimization for tree languages.

2 Optimality Theory

In this section we will formally define the essential notions of OT that are used in the automata constructions later. Let us start by making the notions of optimality theory more precise. In the general case, an OT-system consists of a binary relation **GEN** and a finite set of constraints that are linearly ordered. Constraints may be violated several times. So a constraint should be construed as a function from **GEN** into the natural numbers. Thus an OT-system assigns each candidate pair from **GEN** a sequence of natural numbers. The ordering of the elements of **GEN** that is induced by the OT-system is the lexicographic ordering of these sequences.

Definition 1 *An* OT-system *is a pair* $(\mathbf{GEN}, C)$ *where* **GEN** *is a binary relation and* $C = \langle c_1, \ldots, c_p \rangle, p \in \mathbb{N}$, *is a linearly ordered sequence of functions from* **GEN** *to* $\mathbb{N}$. *Let* $a, b \in \mathbf{GEN}$. *We say* a *is more economical than* b $(a < b)$, *if there is a* $k \leq p$ *such that* $c_k(a) < c_k(b)$ *and for all* $j < k : c_j(a) = c_j(b)$.

Intuitively, an output o is optimal for some input i iff **GEN** relates o to i and o is optimal amongst the possible outputs for i. This is the notion of unidirectional optimality, which is obviously generation-driven. Formally, this reads as follows:

Definition 2 *A form-meaning pair* (f, m) *is* unidirectionally optimal *iff*

1. $(f, m) \in \mathbf{GEN}$,
2. *there is no* (f, m') *such that* $(f, m') < (f, m)$.

Bidirectional optimality reflects the fact that in semantics and pragmatics the relation between input (a form) and output (a meaning) can and perhaps should be regarded as an interplay between parsing optimality and generation optimality. Hence Jäger (2002), formalizing ideas by Blutner (1998, 2000), defines *bidirectional* optimality as follows.

Definition 3 *A form-meaning pair* (f, m) *is* bidirectionally optimal *iff*

1. $(f, m) \in \mathbf{GEN}$,
2. *there is no bidirectionally optimal* (f', m) *such that* $(f', m) < (f, m)$,
3. *there is no bidirectionally optimal* (f, m') *such that* $(f, m') < (f, m)$.

Thus, checking whether a form-meaning pair is bidirectionally optimal requires simultaneous evaluation of form alternatives and meaning alternatives of this pair. This definition is not circular in cases where the ordering of pairs is well-founded. As was shown by Jäger (2002), the ordering of pairs given by the definition of an OT-system is indeed well-founded. Hence bidirectional optimality of OT-systems is not a circular notion.

The type of constraints considered in the literature on finite state OT and also used here is not quite as general as Definition 1 insinuates. Firstly, constraints have to be binary, i.e., a constraint assigns a candidate pair either the number 0 or 1. This restriction is not quite as severe as it may seem. Every constraint with an upper bound on the number of violations can be translated into a sequence of binary constraints.

Secondly, so-called markedness constraints are considered only. A markedness constraint is a constraint that either evaluates solely the input or solely the output.

Definition 4 *Let* $(\mathbf{GEN}, (c_1, \ldots, c_p))$ *be an OT-system.*
A constraint c_j *is an* output markedness constraint *iff* $c_j(i, o) = c_j(i', o)$ *for all* $(i, o), (i', o) \in \mathbf{GEN}$.
A constraint c_j *is an* input markedness constraint *iff* $c_j(i, o) = c_j(i, o')$ *for all* $(i, o), (i, o') \in \mathbf{GEN}$.

To gain a better understanding of how bidirectional optimality is evaluated on markedness constraints in a (finite) OT-system consider this example from Jäger (2002). Suppose $\mathbf{GEN} = \{1, 2, 3\} \times \{1, 2, 3\}$, and we have two constraints which both say "Be small!" One of its instances applies to the input and one to the output. Thus formally we have

- $\mathcal{O} = (\mathbf{GEN}, C)$.
- $\mathbf{GEN} = \{1, 2, 3\} \times \{1, 2, 3\}$.
- $C = \langle c_1, c_2 \rangle$.
- $c_1(\langle i, o \rangle) = i$.
- $c_2(\langle i, o \rangle) = o$.

It follows from the way constraints are evaluated that $\langle i_1, o_1 \rangle <_{\mathcal{O}} \langle i_2, o_2 \rangle$ iff $i_1 \leq i_2, o_1 \leq o_2$, and $\langle i_1, o_1 \rangle \neq \langle i_2, o_2 \rangle$. Now obviously $\langle 1, 1 \rangle$ is bidirectionally optimal since both its input and its output obey the constraints in an optimal way. Accordingly, $\langle 1, 2 \rangle, \langle 2, 1 \rangle, \langle 1, 3 \rangle$, and $\langle 3, 1 \rangle$ are blocked, since they all share a component with a bidirectionally optimal candidate. There are still candidates left which are neither marked as optimal nor as blocked, so we have to repeat this procedure. Amongst the remaining candidates, $\langle 2, 2 \rangle$ is certainly bidirectionally optimal because all of its competitors in either dimension are known to be blocked. This candidate in turn blocks $\langle 2, 3 \rangle, \langle 3, 2 \rangle$. The only remaining candidate, $\langle 3, 3 \rangle$, is again bidirectionally optimal since all its competitors are blocked.[1] This example illustrates the general strategy for the finite case: Find the cheapest input-output pairs in the whole of **GEN** and mark them as bidirectionally optimal. Next mark all candidates that share either the input or the output component (but not both) with one of these bidirectionally optimal candidates as blocked. If there are any candidates left that are neither marked as bidirectionally optimal nor as blocked, repeat the procedure until **GEN** is exhausted.

3 Frank and Satta's Construction

In this section we will discuss the most impressive piece of work on the complexity of OT, Frank and Satta's 1998 construction. This will pave the ground for the extensions and modifications to be presented later on. Before doing so, it is advisable to repeat some prerequisites.

The classes of regular languages and of rational relations are subject to certain *closure properties*.

- Every finite language is regular.
- If L_1 and L_2 are regular languages, then $L_1 \cap L_2, L_1 \cup L_2, L_1 - L_2, L_1 L_2$, and L_1^* are also regular languages.

[1] Bidirectional optimality thus predicts iconicity: the pairing of cheap inputs with cheap outputs is optimal, but also the pairing of expensive inputs with expensive outputs. See Blutner's (1998; 2000) papers for further discussion of this point.

- If R_1 and R_2 are rational relations, then $R_1 \cup R_2, R_1 \circ R_2, R_1^{\cup}, R_1 R_2$, and R_1^* are also rational relations.
- If R is a rational relation, then $Dom(R)$ and $Rg(R)$ (the domain $\{x|\exists y : xRy\}$ and the range $\{y|\exists x : xRy\}$ of R) are regular languages.
- If L_1 and L_2 are regular languages, then $L_1 \times L_2$ and $\mathbf{I}_{L_1}$ are rational relations.

Note that the rational relations are not closed under intersection and complement. Frank and Satta use these closure properties to show that for a significant class of OT-systems, unidirectional optimization is a rational relation provided all building blocks are rational.

Now let us turn our attention to Frank and Satta's construction. They restrict the class of OT-systems in two ways. First, they only consider output markedness constraints. Second, OT constraints in general "count," a given constraint may be violated arbitrarily many times. Frank and Satta restrict attention to binary constraints, i.e. constraints c with the property $Rg(c) = \{0, 1\}$. OT-systems which are not binary but have an upper limit for the number of constraint violations are implicitly covered; a constraint c that can be violated at most n times can be represented by n binary constraints of the form "Violate c less than i times" for $1 \leq i \leq n$. The ranking of these new constraints is inessential for the induced ordering relation.

As said, Frank and Satta restrict attention to binary output markedness constraints. Obviously, these can be represented as languages over the output alphabet, namely as the set of outputs that obey them.

The central part of their construction is an operation called *conditional intersection* that combines a relation with a language. Karttunen (1998) calls this operation *lenient composition*, and we will follow this terminology.

Definition 5 (Lenient composition) *Let R be a relation and $L \subseteq Rg(R)$. The* lenient composition *$R \uparrow L$ of R with L is defined as*

$$R \uparrow L \doteq (R \circ \mathbf{I}_L) \cup (\mathbf{I}_{Dom(R) - Dom(R \circ \mathbf{I}_L)} \circ R)$$

By applying the definitions, it is easy to see that $\langle x, y\rangle \in R \uparrow L$ iff xRy and either $y \in L$ or there is no $z \in L$ such that xRz. In other words, $\{y|\langle x, y\rangle \in R \uparrow L\}$ is the set of ys that are related to x by R, and that are optimal with respect to the constraint L. Furthermore, it follows from the closure properties of regular sets that $R \uparrow L$ is a rational relation provided R is rational and L is a regular language.

Unidirectional optimality can now be implemented in a straightforward way, namely by successively leniently composing the (binary markedness) constraints of an OT-system with **GEN**.

Theorem 1 (Frank and Satta) *Let $\mathcal{O} = \langle \mathbf{GEN}, C\rangle$ with $C = \langle c_1, \ldots, c_p\rangle$ be an OT-system such that C solely consists of binary output markedness constraints. Then $\langle i, o\rangle$ is unidirectionally optimal iff $\langle i, o\rangle \in \mathbf{GEN} \uparrow c_1 \cdots \uparrow c_p$.*

The proof of this theorem is obvious from the definitions. Crucially, it follows that unidirectional optimality is a rational relation provided **GEN** is rational and all constraints are regular languages.

4 Extension to Bidirectionality

The informal reasoning that was used to calculate the set of bidirectionally optimal elements in the 3×3–example above can be generalized to a rigorous algorithm. However, such an evaluation strategy is confined to OT-systems with a finite **GEN**. The finite state construction in Jäger (2002) follows the same recursive strategy, but it is also applicable to systems with an infinite generator, provided all building blocks are finite state implementable.

The first step inside the evaluation strategy used above amounts to finding the globally optimal input-output pairs in the whole of **GEN**. As in Frank and Satta's construction, this global optimization is achieved by successively optimizing with respect to the constraints in C in the order of their ranking. Jäger's 2002 construction is also restricted to binary markedness constraints. If we want to leniently compose **GEN** with a binary input markedness constraint, we need a mirror image of Frank and Satta's lenient composition. Thus backward lenient composition is defined as

$$R \downarrow L \doteq (\mathbf{I}_L \circ R) \cup (R \circ \mathbf{I}_{Rg(R)-Rg(\mathbf{I}_L \circ R)})$$

Furthermore, for reasons that will become clear immediately, in bidirectional optimality it is not sufficient to consider the best outputs for a given input, but we have to look for the best input-output pairs in a global way. Thus bidirectional lenient composition is defined in the following way:

Definition 6 (Bidirectional Lenient Composition) *Let $\mathcal{O} = \langle \mathbf{GEN}, C \rangle$ be an OT-system with $C = \langle c_1, \ldots, c_p \rangle$, and c_i be a binary markedness constraint.*

$$R \Uparrow c_i \doteq \begin{cases} R \circ \mathbf{I}_{Rg((\{\varepsilon\} \times Rg(R)) \uparrow c_i)} \\ \textit{if } c_i \textit{ is an output markedness constraint} \\ \\ \mathbf{I}_{Dom((Dom(R) \times \{\varepsilon\}) \downarrow c_i)} \circ R \\ \textit{else} \end{cases}$$

Let us look at this construction in detail. Suppose c_i is an output markedness constraint. $\{\varepsilon\} \times Rg(R)$ is a relation that relates the empty string to any possible output of R. Leniently composing this relation with c_i leads to a relation that relates the empty string to those possible outputs of R that are optimal with respect to c_i. So if c_i is fulfilled by some output of R, this relation is just $\{\varepsilon\} \times (Rg(R) \cap c_i)$. If no output of R obeys c_i, the relation is just $\{\varepsilon\} \times Rg(R)$. In either way, $Rg((\{\varepsilon\} \times Rg(R)) \uparrow c_i)$ is the set of outputs of R that are optimal with respect to c_i. Since c_i only evaluates outputs, $R \Uparrow c_i$ is thus the set of $\langle i, o \rangle \in R$ that are optimal with respect to c_i. The same holds *ceteris paribus* if c_i is an input markedness constraint.

Like Frank and Satta's operation, bidirectional lenient composition only makes use of finite state techniques. It follows directly from the closure properties of regular languages and rational relations that $R \Uparrow c_i$ is a rational relation provided R is rational and c_i is a regular language.

Note that a certain input-output pair may be evaluated as sub-optimal according to this construction even if it neither shares the input component nor the output component with any better candidate. So while Frank and Satta's lenient composition operates pointwise for each input, bidirectional lenient composition is global.

Fact 1 *Let $\mathcal{O} = \langle \mathbf{GEN}, C \rangle$ be an OT-system (with binary markedness constraints only), where $C = \langle c_1, \ldots, c_p \rangle$, and let $R \subseteq \mathbf{GEN}$. Then*

$$\langle i, o \rangle \in R \Uparrow c_1 \cdots \Uparrow c_p$$

iff $\langle i, o \rangle \in R$, and there are no i', o' with $\langle i', o' \rangle \in R$ and $\langle i', o' \rangle < \langle i, o \rangle$.

The proof of this fact can be found in Jäger (2002).

For simplicity, the notation R^C as is used as shorthand for $\{x \in R | \neg \exists y \in R : y <_{\mathcal{O}} x\}$, where $\mathcal{O}$ is some OT-system with C as its constraint component. Intuitively, this operation picks out the globally optimal set of input-output pairs from R. The above fact states that $R^C = R \Uparrow c_1 \cdots \Uparrow c_p$ (where $C = \langle c_1, \ldots, c_p \rangle$). Note that R^C is a rational relation if R is rational and all constraints in C are regular languages.

After finding the globally optimal input-output pairs OPT from **GEN**, we have to reevaluate those pairs from **GEN** that neither share the input component nor the output component with any globally optimal input-output pair. These pairs form the set

$$\mathbf{I}_{Dom(\mathbf{GEN}) - Dom(OPT)} \circ \mathbf{GEN} \circ \mathbf{I}_{Rg(\mathbf{GEN}) - Rg(OPT)}$$

Optimization has to be applied to this set as well, and this procedure has to be repeated until all elements of **GEN** are either evaluated as optimal or as blocked. This leads to the following recursive definition of bidirectional optimality:

$$\begin{aligned} OPT_0 &= \emptyset \\ OPT_{n+1} &= (\mathbf{I}_{Dom(\mathbf{GEN}) - Dom(OPT_n)} \circ \mathbf{GEN} \circ \mathbf{I}_{Rg(\mathbf{GEN}) - Rg(OPT_n)})^C \end{aligned}$$

Fact 2 *Let $\mathcal{O} = \langle \mathbf{GEN}, C \rangle$ be an OT-system such that all c_i in C are binary markedness constraints. If* **GEN** *is a rational relation and all constraints can be represented as regular languages, OPT_n is a rational relation for each n.*

Proof: Immediately from the construction of OPT and $(\cdot)^C$ and the closure conditions on regular languages/rational relations. ⊣

Even if **GEN** is infinite, there are only finitely many possible patterns of constraint violations if we are dealing with OT-systems comprising only binary constraints. This guarantees that bidirectional optimization with binary markedness constraints remains within the reach of finite state techniques.

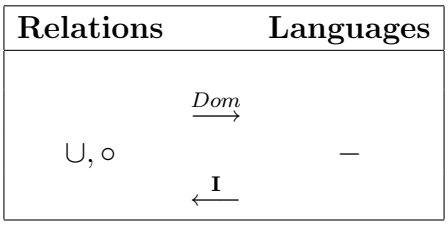


Figure 1: Closure conditions needed for undirectional optimality

Lemma 1 *Let $\mathcal{O} = \langle \mathbf{GEN}, C\rangle$ be an OT-system with $C = \langle c_1, \ldots, c_p\rangle$, where all c_i are binary markedness constraints. Then $\langle i, o\rangle$ is bidirectionally optimal iff $\langle i, o\rangle \in OPT_{2^p}$.*

Proof: See Jäger (2002). ⊣

This leads directly to the central result of Jäger (2002), namely the bidirectional counterpart of Frank and Satta's 1998 result.

Theorem 2 *Let $\mathcal{O} = \langle \mathbf{GEN}, C\rangle$ be an OT-system with $C = \langle c_1, \ldots, c_p\rangle$, where all c_i are binary markedness constraints. Furthermore, let* **GEN** *be a rational relation and let all c_i be regular languages. Then the set of bidirectionally optimal elements of* **GEN** *is a rational relation.*

Proof: Directly from Fact 2, Lemma 1, and the closure properties of regular languages and rational relations. ⊣

It is important to appreciate that neither Frank and Satta (1998) in their proof for undirectional optimization, nor Jäger (2002) in his proof for bidirectional optimization, make any use of the specific properties of regular languages and rational relations. All that is needed are certain algebraic closure properties of the classes of languages and relations. The closure operations that are needed in the proof by Frank and Satta are given in Figure 1.

Jäger's proof for bidirectional optimization is also entirely algebraic, but it makes use of more closure properties. Crucially, it requires the Cartesian product of two regular languages to be a rational relations. Figure 2 shows the closure operations that are used in that proof.

5 Tree Automata

5.1 The Idea

A finite state automaton is a machine that processes the items in a string, one at a time, from left to right. If such a machine is for instance fed with the string

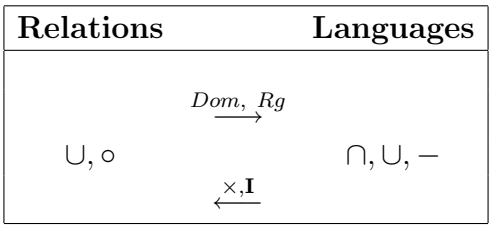
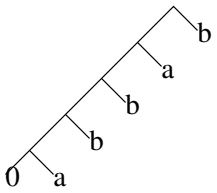


Figure 2: Closure conditions needed for bidirectional optimality

abbab

it starts with processing the first a, thereby possibly changing its state, proceeds to the first b etc., until it reaches the end of the string.

A string can equivalently be conceived as a uniformly right-branching (or uniformly left-branching) tree. The above example thus becomes



where "0" is a marker for the empty tree. A string automaton processing the above string can equivalently be conceived as a machine that processes this kind of tree in a bottom-up fashion.

A bottom-up "finite state tree automaton" is a generalization of this idea to arbitrary trees. It can process various nodes of the same tree in parallel independently of each other. If all daughters of a given node are processed, it can move to the mother node. This means that it can start at any leaf without special preconditions. A tree as a whole is processed if the automaton reaches the root node.

In the remainder of this section, we will spell out this intuition in formal terms.

5.2 Regular Tree Grammars

A *ranked alphabet* (or *ranked operator domain*) Σ is an indexed family $\langle \Sigma_n \rangle_{n \in \mathbb{N}}$ of disjoint sets. A symbol f in Σ_n is called an *operator of rank* n. If $n = 0$, then f is also called a constant. For a ranked alphabet Σ, the set of *trees over* Σ (or Σ-*trees* or *terms over* Σ), denoted T_Σ is the smallest set of strings over $\Sigma \cup \{(,)\}$ such that $\Sigma_0 \subseteq T_\Sigma$ and, for $n \geq 1$, if $f \in \Sigma_n$ and $t_1, \ldots, t_n \in T_\Sigma$, then $f(t_1, \ldots, t_n) \in T_\Sigma$. A subset of T_Σ is called a *tree language* over Σ.

Any set M can be interpreted as a signature in which all of the elements of M are constants. Thus $T_{\Sigma \cup M}$ denotes the set of trees over Σ and M. Let $X = \{x_1, x_2, x_3, \ldots\}$ be

an infinite set (of variables). Then $T_\Sigma(x_1, \ldots, x_n) = T_{\Sigma \cup \{x_1, \ldots, x_n\}}$ denotes the set of trees over Σ and additional variables $\{x_1, \ldots, x_n\}$. From the point of view of a signature, a "variable" is a constant. If $t \in T_\Sigma(X)$ then we also write $t(x_1, \ldots, x_n)$ to indicate that the variables of t are a subset of the set $\{x_1, \ldots, x_n\}$. Let Σ and Ω be two signatures, let $t(x_1, \ldots, x_n) \in T_\Sigma(x_1, \ldots, x_n)$, and let $t_1, \ldots, t_n \in T_\Omega$. Then $t(t_1, \ldots, t_n) \in T_{\Sigma \cup \Omega}$ is the result of simultaneously replacing each occurrence of x_j in $t(x_1, \ldots, x_n)$ by t_j (with $1 \leq j \leq n$).

Now we define the notion of a regular tree grammar.

Definition 7 *A* regular tree grammar *is a quadruple* $G = (\Sigma, \mathcal{F}, S, P)$ *where*

Σ	*is a finite ranked alphabet of* terminals*,*
$\mathcal{F}$	*is a finite set of* nonterminals*, disjoint with* Σ*,*
$S \in \mathcal{F}$	*is the* start symbol*, and*
P	*is a finite set of productions (or rules) of the form* $F \to t$*, where* $F \in \mathcal{F}$ *and* $t \in T_{\Sigma \cup \mathcal{F}}$*.*

For a regular tree grammar $G = (\Sigma, \mathcal{F}, S, P)$ the derivation relation is given as follows. Let $s_1, s_2 \in T_{\Sigma \cup \mathcal{F}}$. We say $s_1 \Rightarrow s_2$ if and only if there is a production $F \to t$ and F is a leaf node of s_1. Tree s_2 is obtained from s_1 by replacing F with the tree t. As usual, $\stackrel{*}{\Rightarrow}$ stands for the reflexive-transitive closure of $\Rightarrow$. For a regular tree grammar G, we define $L(G) = \{t \in T_\Sigma \mid S \stackrel{*}{\Rightarrow} t\}$. $L(G)$ is called the *tree language* generated by G.

5.3 Tree Automata and Tree Transducers

For regular tree languages there exists an automaton model that corresponds to finite state automata for regular string languages. Let Σ be a signature. A *frontier-to-root tree automaton* is a triple $\mathcal{A} = (Q, F, \delta)$ where Q is a finite set of *states*, $F \subseteq Q$ a set of *final* states and δ is a finite transition relation. Each transition has the form

$$f(q_1, \ldots, q_n) \to q$$

where $f \in \Sigma_n$, $q, q_1, \ldots, q_n \in Q$. On an intuitive level, a frontier-to-root tree automaton labels the nodes in a tree with states starting from the leaves and going to the root. Suppose n is a node in the tree and f is the k-ary function symbol at node n and the k daughters of n are already labelled with states $q_1, \ldots, q_k$, and furthermore $f(q_1, \ldots, q_k) \to q$ is a transition of $\mathcal{A}$, then node n can be labelled with state q. A tree is accepted if the root can be labelled with a final state.

We will now report some results about the theory of regular tree languages. For more information, consult the work by Gécseg and Steinby (1984, 1997). A tree language L is regular if and only if there is a tree automaton that accepts L. Regular tree languages are closed under union, intersection, and relative complement. There are corresponding constructions for tree automata. And finally, for every tree automaton accepting language L there exists a tree automaton also accepting L which has a single final state.

Tree automata can be generalized to automata that transform one tree into another, so-called tree transducers. The following exposition on tree transduction is taken from (Gécseg

and Steinby, 1997). Let Σ and Ω be two signatures. A binary relation $\tau \subseteq T_\Sigma \times T_\Omega$ is called a *tree transformation*. A pair $(s,t) \in \tau$ is interpreted to mean that τ may transform s into t. We can speak of compositions, inverses, domains, and ranges of tree transformations as those of binary relations. We will now define frontier-to-root tree transducers.

Definition 8 *A* frontier-to-root tree transducer *(or F-transducer) consists of a quintuple* $\mathcal{A} = (\Sigma, \Omega, Q, P, F)$ *where* Σ *and* Ω *are signatures;* Q *is a finite set of* states*, each element of* Q *is a unary function;* $F \subseteq Q$ *is the set of* final states*; and* P *is a finite set of* productions *of the following type:*

$$f(q_1(x_1), \ldots, q_m(x_m)) \to q(t(x_1, \ldots, x_m))$$

where $f \in \Sigma_m$, $q_1, \ldots, q_m, q \in Q$, $t(x_1, \ldots, x_m) \in T_\Omega(x_1, \ldots, x_m)$.

The transformation induced by an F-transducer is defined as follow. We write QT_Ω for the set $\{q(t) \mid q \in Q, t \in T_\Omega\}$ and regard QT_Ω as a set of constants. Let $s, t \in T_{\Sigma \cup QT_\Omega}$ be two trees. It is said that t can be obtained by a direct derivation from s in $\mathcal{A}$ iff t can be obtained from s by replacing an occurrence of a subtree $f(q_1(t_1), \ldots, q_m(t_m))$ (with $f \in \Sigma_m, q_1, \ldots, q_m \in Q, t_1, \ldots, t_m \in T_\Omega$) in s by $q(t(t_1, \ldots, t_m))$, where $f(q_1(x_1), \ldots, q_m(x_m)) \to q(t(x_1, \ldots, x_m))$ is a production from P. If s directly derives t in $\mathcal{A}$ then we write $s \Rightarrow_\mathcal{A} t$. The reflexive transitive closure $s \Rightarrow^*_\mathcal{A} t$ is the derivation relation.

Intuitively, an F-transducer traverses a tree s from the leaves to the root rewriting it at the same time. In a single derivation step we consider a node n in s with label f where all the daughter nodes are already transformed into trees of T_Ω and each daughter node is in some state q_i. Then we replace the subtree of node n with the tree t from the production where the place holder variables of t are replaced by the trees of the daughter nodes of n. The root of this subtree is put into state q.

The relation

$$\tau_\mathcal{A} = \{(s,t) \mid s \in T_\Sigma, t \in T_\Omega, s \Rightarrow^*_\mathcal{A} q(t) \text{ for some } q \in F\}$$

is the transformation relation induced by $\mathcal{A}$. A relation $\tau \subseteq T_\Sigma \times T_\Omega$ is an F-*transformation* if there exists an F-transducer $\mathcal{A}$ such that $\tau = \tau_\mathcal{A}$. For a tree language $L \subseteq T_\Sigma$ we define $\mathcal{A}(L) = \{t \in T_\Omega \mid \exists s \in L \text{ with } (s,t) \in \tau_\mathcal{A}\}$.

A production $f(q_1(x_1), \ldots, q_m(x_m)) \to q(t(x_1, \ldots, x_m))$ is called *linear* if each variable $x_1, \ldots, x_m$ occurs at most once in t. An F-transducer is linear if each production is linear. We denote a linear F-transducer by LF-transducer. We will make use of the following results about LF-transducers.

Fact 3 *LF-transducers are closed under composition.*
The class of regular tree languages is closed under LF-tranductions.
The domain and range of an LF-transducer is a regular tree language.
For every regular tree language L *there is an LF-transducer* ι *such that* $Dom(\iota) = Rng(\iota) = L$ *and* ι *is the identity on* L.

These results can be found, e.g., in (Gécseg and Steinby, 1984, 1997). If $\mathcal{A}_1$ and $\mathcal{A}_2$ are two LF-transducers, we write $\mathcal{A}_1 \circ \mathcal{A}_2$ for the composition of first $\mathcal{A}_1$ and then $\mathcal{A}_2$. Let L_1 and L_2 be two tree languages. A binary relation $R \subseteq L_1 \times L_2$ is called *rational* if there exists an LF-transducer $\mathcal{A}$ such that $\tau_{\mathcal{A}} = R$.

LF-transducers define relations between tree languages where there is a strong connection between the input trees and the output trees. Obviously, an output tree is constructed on the base of a structural decomposition of the input tree. As stated in the introduction, we also need an automaton or a transducer representation of the Cartesian product of arbitrary regular tree languages. In a Cartesian product, every element of the input tree language is related to every element of the output tree language. Hence there is in general no structural relation between an input tree and an output tree. Therefore LF-transducers are not capable of defining the Cartesian product of two regular tree languages.

6 Unidirectional Optimality of Tree Languages

The second direction to extend the original approach by Frank and Satta consists of the transition from string languages to tree languages. The original approach provided an automaton theoretic formalization of OT for the original domain of application of OT, namely phonology. But, as mentioned in the introduction, OT is nowadays used as a formal framework in syntax, semantics and pragmatics. In these areas of linguistics, it is trees rather than strings that form the underlying datastructures linguists reason about. Hence, if one wants to provide an automaton theoretic formalization of OT, the relevant languages to be taken as base have to be *tree* languages, and the automata have to be *tree* automata.

Wartena (2000) was the first to provide a tree automata theoretic formalization of OT. It is based on the observation that closure properties of regular string languages extend to regular tree languages. And finite state transducers have their counterparts in LF-transducers. Since all of the closure properties needed for unidirectional optimality (see Figure 1) do indeed exist, as was shown in the last section, Wartena (2000) proved the generalization of the result by Frank and Satta for regular tree languages.

Theorem 3 (Wartena) *Let $\mathcal{O} = \langle \mathbf{GEN}, C \rangle$ be an OT-system with $C = \langle c_1, \ldots, c_p \rangle$, where all c_i are binary output markedness constraints. Furthermore, let* **GEN** *be a rational tree relation and let all c_i be regular tree languages. Then the set of bidirectionally optimal elements of* **GEN** *is a rational tree relation.*

Proof: Parallel to Frank and Satta's proof. ⊣

Observing that there are certain non-regular phenomena in some natural languages (see, e.g., (Shieber, 1985)) Kepser and Mönnich (2003) extend the result by Wartena to linear context-free tree languages. In their work, the generator is split into a source for input trees defined by means of a linear context-free tree grammar and a relation between input trees and output trees defined by a linear tree transducer. Constraints are defined

by regular tree languages. Their modularity result is based on closure properties of linear context-free tree languages also shown in that paper.

A *context-free tree grammar* (CFTG) is a quintuple $\Gamma = \langle \Sigma, \mathcal{F}, S, X, P \rangle$ where Σ and F are ranked alphabets of *terminals* and *nonterminals*, $S \in \mathcal{F}$ is the start symbol, X is a countable set of variables, and P is a finite set of productions. Each production $p \in P$ is of the form $F(x_1, \ldots, x_n) \rightarrow t$ for some $n \in \mathbb{N}$, where $F \in \mathcal{F}_n, x_1, \ldots, x_n \in X$ and t is a tree over $\Sigma \cup \mathcal{F}, \{x_1, \ldots, x_n\}$. The grammar is *linear* iff each variable occurs at most once in the right hand side of each rule. Intuitively, an application of a rule of the form $F(x_1, \ldots, x_n) \rightarrow t$ "rewrites" a tree rooted in F as the tree t with its respective variables substituted by F's daughters.

A regular tree grammar is a special case of a CFTG, namely the one in which all nonterminals are of arity 0, i.e., constants. If $\mathcal{F}_n$ is non-empty for some $n \neq 0$, that is, if we allow the *nonterminals* to be parameterized by variables, however, the situation changes. CFTGs in general are capable of generating sets of structures, the *yields* of which belong to the subclass of context-sensitive languages known as the *indexed* languages.

Linear context-free tree languages are closed under LF-transductions and under intersection with regular tree languages, as was shown by Kepser and Mönnich (2003). Therefore Kepser and Mönnich provide a formalization of OT where the generator is defined by an LF-transducer on a linear context-free tree language as input.

Theorem 4 *Let $\mathcal{O} = \langle \mathbf{GEN}, C \rangle$ be an OT-system with $C = \langle c_1, \ldots, c_p \rangle$, where all c_i are binary output markedness constraints. Furthermore, let* **GEN** *be defined by an LF-transduction on a linear context-free tree language as input, and let all c_i be regular tree languages. Then the set of optimal output elements is a linear context-free tree language that can be computed by finite state techniques.*

The proof of this theorem can be found in (Kepser and Mönnich, 2003).

7 Bidirectional Optimality of Regular Tree Languages

As mentioned above, it is not guaranteed that the Cartesian product of two regular tree languages is a rational tree relations. Therefore Jäger's (2002) proof cannot be lifted to tree languages as straightforwardly as Frank and Satta's proof. Nonetheless, bidirectional optimization can be implemented by means of finite state tree automata. There are two techniques how the obstacle (of non-closure under Cartesian product) can be overcome: We can generalize the notion of tree transduction, or we can modify the implementation of bidirectional lenient composition. Both strategies are successful.

Let us start with the first. The Cartesian product of two (or more) regular tree languages can be defined by means of *tree tuple automata*. Comon et al. (1997, Sect. 3.2) describe three classes of tree tuple automata two of which can be used for the definition of a Cartesian product. The simplest class, which suffices already for our purposes, is based on pairs of automata. Let L_1 and L_2 be two regular tree languages, and let $\mathcal{A}_1$ be a bottom-up tree automaton accepting L_1, and $\mathcal{A}_2$ a bottom-up tree automaton accepting

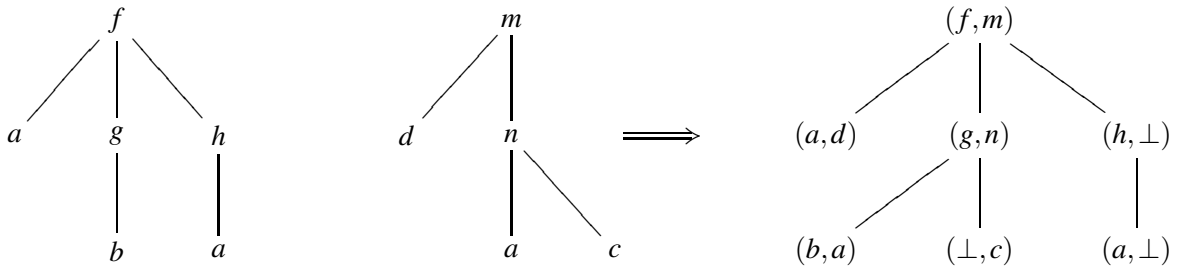


Figure 3: Coding two trees in a single one.

L_2. Define for any two trees s, t that $(s,t) \in (\mathcal{A}_1, \mathcal{A}_2)$ iff $s \in \mathcal{A}_1$ and $t \in \mathcal{A}_2$. Then, obviously, $(\mathcal{A}_1, \mathcal{A}_2)$ defines the relation $L_1 \times L_2$. This relation is called $Rec_\times$ in (Comon et al., 1997).

A more powerful definition of tree tuple automata is given, if the relation is expressed by a single automaton instead of a pair of automata. To make this approach work, two trees have to be coded in a single one. Let Σ and Ω be two signatures. We define trees with labels as pairs of $(\Sigma \cup \{\bot\} \times \Omega \cup \{\bot\})$ where $\bot$ is a new (padding) symbol not contained in Σ or Ω. For a pair of trees $(s,t) \in (T_\Sigma \times T_\Omega)$ we define inductively their coding $[s,t]$ by:

$$[f(t_1,\ldots,t_n), g(u_1,\ldots,u_m)] = \begin{cases} (f,g)([t_1,u_1],\ldots,[t_m,u_m],[t_{m+1},\bot],\ldots,[t_n,\bot]) & \text{if } n \geq m \\ (f,g)([t_1,u_1],\ldots,[t_n,u_n],[\bot,u_{n+1}],\ldots,[\bot,u_m]) & \text{if } m \geq n \end{cases}$$

Basically, the union of the tree domains is constructed and the nodes are labelled with pairs of labels. If one tree lacks a branch, then the padding symbol $\bot$ is used. See Figure 3 for an example.

Now, *Rec* is the set of relations $R \subseteq T_\Sigma \times T_\Omega$ such that $\{[s,t] \mid (s,t) \in R\}$ is accepted by a bottom-up tree automaton on the signature $(\Sigma \cup \{\bot\} \times \Omega \cup \{\bot\})$.

We quote some results on tree tuple automata (see Comon et al., 1997). $Rec_\times$ is strictly included in *Rec*. $Rec_\times$ and *Rec* are closed under union, intersection and complement. Also, it is straightforward to see that the identity relation on a regular tree language is included in *Rec*. Likewise, it is obvious that the domain and the range of a relation in *Rec* is a regular tree language. Thus the class of regular tree languages and the class of relations *Rec* have all the closure properties that are used in the proof for bidirectional optimization. Therefore we get

Theorem 5 *Let $\mathcal{O} = \langle \mathbf{GEN}, C \rangle$ be an OT-system with $C = \langle c_1, \ldots, c_p \rangle$, where all c_i are binary markedness constraints. Furthermore, let* **GEN** $\in$ *Rec, and let all c_i be regular tree languages. Then the set of bidirectionally optimal elements of* **GEN** *is a tree relation in Rec.*

Proof: Completely parallel to the proof of the corresponding theorem in Jäger (2002). $\dashv$

From a processing point of view, this construction is not fully satisfactory, because it only allows to check efficiently whether a given input-output pair is related via bidirectional optimality. It does not directly lead to an efficient procedure to compute the set of bidirectionally optimal outputs for a given input, say. In this sense, the rational relations, i.e., the relations that directly correspond to LF-transducers are more interesting from a computational point of view than the more abstract class *Rec*. But the construction of the Cartesian product can be used in the construction of bidirectional lenient composition in combination with LF-transducers. Therefore **GEN** can be defined as a rational tree relation and we obtain Theorem 6 (for details, see (Kepser, 2004)).

Rather than explaining the details of the construction using *Rec* for the Cartesian product in bidirectional lenient composition, let us thus reconsider why a lifting of the proof from Jäger (2002) to rational tree relations failed. There is no guarantee that the Cartesian product of two regular tree languages is a rational tree relation. However, all that is actually needed in the proof is the construction $\{\varepsilon\} \times L$ and $L \times \{\varepsilon\}$ for a given regular language L, not the Cartesian product between arbitrary languages. Even further, it does not matter for the construction what language the first argument of this product is — any non-empty language would do. So we actually only need closure under the operations $L \mapsto A \times L$ and $L \mapsto L \times A$, where A is some arbitrary given non-empty language. The latter of these two operations is actually definable as an LF-transducer. Let $*$ be an arbitrary new symbol. The LF-transducer in question has just one state, q, and for each function symbol f in the regular tree language in question, it contains the production

$$f(q(*), \ldots, q(*)) \rightarrow q(*)$$

This transducers maps any tree to the language $\{*\}$. Composing it with the identity transducer on a rational language L thus yields a transducer that implements the operation $L \times \{*\}$.

Unfortunately, this construction does not work backwards, i.e., there is no LF-transducer that would implement the operation $L \mapsto \{*\} \times L$. This operation is used to define bidirectional lenient composition for output markedness constraints. However, this can be redefined in parallel to the case for input markedness constraints, simply by using the range, rather than the domain, of the input relation. The revised version of bidirectional lenient composition thus becomes

Definition 9

$$R \Uparrow c_i \doteq \begin{cases} R \circ \mathbf{I}_{Dom((Rg(R)\times\{*\})\uparrow c_i)} \\ \textit{if } c_i \textit{ is an output markedness constraint} \\ \\ \mathbf{I}_{Dom((Dom(R)\times\{*\})\downarrow c_i)} \circ R \\ \textit{else} \end{cases}$$

It now directly follows from the closure properties of regular tree languages and rational tree relations that the bidirectional lenient composition of a rational tree relation with a regular tree language is again a rational tree relation. This in turn directly entails

Theorem 6 *Let $\mathcal{O} = \langle \mathbf{GEN}, C \rangle$ be an OT-system with $C = \langle c_1, \ldots, c_p \rangle$, where all c_i are binary markedness constraints. Furthermore, let* **GEN** *be a rational tree relation, and let all c_i be regular tree languages. Then the set of bidirectionally optimal elements of* **GEN** *is a rational tree relation.*

Proof: Completely parallel to the proof of the corresponding theorem in Jäger (2002), except that the definition of bidirectional lenient composition given above is used. ⊣

8 Conclusion

This paper reviews some results on the automata theoretic complexity of unidirectional and bidirectional OT. Frank and Satta (1998) showed that regular string languages and rational string relations are closed under unidirectional optimization. Various authors extended this result for bidirectional optimization of string languages and to unidirectional optimization of tree languages. We presented two results that merge these two lines of research. We gave two constructions for bidirectional optimization of regular tree languages. While this class of tree languages is arguably still too limited to implement syntactic and semantic generalizations in its entirety, it is already expressive enough to deal with substantial fragments in a rigorous way.

References

Blutner, Reinhard. 1998. Lexical pragmatics. *Journal of Sematics* 15:115–162.

Blutner, Reinhard. 2000. Some aspects of optimality in natural language interpretation. *Journal of Semantics* 17:189–216.

Comon, Hubert, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 1997. Tree automata techniques and applications. Available at: `http://www.grappa.univ-lille3.fr/tata`. Release October, 1st 2002.

Frank, Robert and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24:307–315.

Gécseg, Ferenc and Magnus Steinby. 1984. *Tree Automata*. Budapest: Akademiai Kiado.

Gécseg, Ferenc and Magnus Steinby. 1997. Tree languages. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages, Vol 3: Beyond Words*, pages 1–68. Springer-Verlag.

Jäger, Gerhard. 2002. Some notes on the formal properties of bidirectional optimality theory. *Journal of Logic, Language, and Information* 11:427–451.

Karttunen, Lauri. 1998. The proper treatment of optimality in computational phonology. manuscript. Xerox Research Centre Europe.

Kepser, Stephan. 2004. Bidirectional optimality for regular tree languages. In G. Jäger, P. Monachesi, G. Penn, and S. Wintner, eds., *Proceedings of Formal Grammar 2004*, pages 63–76. ESSLLI 2004.

Kepser, Stephan and Uwe Mönnich. 2003. A note on the complexity of optimality theory. In G. Scollo and A. Nijholt, eds., *Proceedings Algebraic Methods in Language Processing (AMiLP-3)*, pages 153–166.

Shieber, Stuart. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8:333–343.

Wartena, Christian. 2000. A note on the complexity of optimality systems. In R. Blutner and G. Jäger, eds., *Studies in Optimality Theory*, pages 64–72. University of Potsdam. Also available at Rutgers Optimality Archive as ROA 385-03100.